



# *MSM*

## *Pocket Guide*



MICRONETICS



**MSM**

**MICRONETICS STANDARD MUMPS**

**POCKET GUIDE**

**Revised**

**April 1, 1993**

**—— Version 4.0 ——**

**COPYRIGHT © 1984-1993**

## **DISCLAIMER**

Micronetics Design Corporation makes no representations or warranties with respect to this manual. Further, Micronetics Design Corporation reserves the right to make changes in the specification of the product described within this manual and without obligation of Micronetics Design Corporation to notify any person of such revision or changes.

The software described in this document is furnished under a license by Micronetics Design Corporation and may only be used or copied in accordance with the terms of such license.

This manual is copyrighted. All rights are reserved. This document may not, in whole or part, be copied, photocopied, reproduced, translated or reduced to any electronic medium or machine readable form without prior consent, in writing, from Micronetics Design Corporation.

Copyright © 1984-1993  
Micronetics Design Corporation  
1375 Piccard Drive  
Rockville, Maryland 20850  
301-258-2605  
Telex 46-9677  
Facsimile 301-840-8943

## **Table of Contents**

|           |  |
|-----------|--|
| <b>v</b>  | <b>Documentation Conventions</b>         |
| <b>1</b>  | <b>MSM Control Characters</b>            |
| <b>2</b>  | <b>MSM Operators</b>                     |
| <b>4</b>  | <b>MSM Pattern Match Characters</b>      |
| <b>5</b>  | <b>MSM Commands</b>                      |
| <b>13</b> | <b>MSM Functions</b>                     |
| <b>19</b> | <b>MSM Special Variables</b>             |
| <b>22</b> | <b>MSM Structured System Variables</b>   |
| <b>23</b> | <b>MSM Device Numbers</b>                |
| <b>24</b> | <b>MSM Device Parameters</b>             |
| <b>36</b> | <b>MSM Full Screen Editor</b>            |
| <b>38</b> | <b>MSM CUA Full Screen Editor</b>        |
| <b>40</b> | <b>MSM Error Messages</b>                |
| <b>47</b> | <b>MSM Error Numbers</b>                 |
| <b>51</b> | <b>ANSI 3.64-1979 Mnemonic Namespace</b> |
| <b>53</b> | <b>ZWINTERM Mnemonic Namespace</b>       |
| <b>57</b> | <b>ASCII Character Set</b>               |
| <b>61</b> | <b>Summary of Terminology</b>            |

## Acknowledgement

Micronetics Standard MUMPS (MSM) is a complete implementation, with extensions, of the ANSI Standard Specification (X11.1-1990) for the Massachusetts General Hospital Utility Multi Programming System (MUMPS). MUMPS was developed by the Laboratory of Computer Science at Massachusetts General Hospital under grant number HS00240 from the National Center for Health Services Research and Development. MUMPS is a trademark of the Massachusetts General Hospital.

## Documentation Conventions

The following conventions are used throughout this booklet.

| Convention                 | Description   |
|----------------------------|---|
| <code>RET</code>           | The Carriage Return Key (normally labeled RETURN, RET, ENTER, etc.)   |
| <code>CTRL/X</code>        | The Control Key pressed at the same time as the 'X' key.  |
| val <code>[SP]</code> val  | Two values separated by one and only one space.   |
| <code>&lt;ERROR&gt;</code> | An error message produced by MSM. Refer to the section titled <i>MSM Error Messages</i> beginning on page 36 of this guide for a complete list of error messages. |
| <code>(parm)</code>        | A parameter that is shown in parentheses is considered part of the statement and must be typed with the parentheses.  |
| <code>{parm}</code>        | An optional parameter or an optional part of a statement. When entering the optional part, the braces are not included.   |

| Convention  | Description   |
|-------------|---|
| • • •       | The series of items repeats a user specified number of times.                       |
| <b>BOLD</b> | Items in a dialogue shown in <b>bold</b> indicate a user response.                  |
| •<br>•<br>• | Shows a break in a list where consecutive lines have been omitted.                  |
| ""          | Items shown in double quotes are string literals and are entered with the quotes.   |
| ' '         | Items shown in single quotes are user responses and are entered without the quotes. |

Throughout this booklet, the syntax of the language and the programming examples are shown with the minimum number of spaces required to be correct. The current ANSI standard allows additional spaces to be added between commands to improve readability and format. Everywhere within this manual where one space is shown between commands, one or more spaces may be used.

## MSM Control Characters

| Character  | Function Performed  |
|--|---|
| Backspace  | Deletes the last character entered.   |
| <span style="border: 1px solid black; border-radius: 5px; padding: 2px;">DEL</span>    | Deletes the last character entered.   |
| <span style="border: 1px solid black; border-radius: 5px; padding: 2px;">ESC</span>    | Terminates input or invokes ESCAPE processing.  |
| <span style="border: 1px solid black; border-radius: 5px; padding: 2px;">RET</span>    | Terminates current input operation.   |
| <span style="border: 1px solid black; border-radius: 5px; padding: 2px;">TAB</span>    | Moves cursor to next tab stop. Tab stops are every 8 positions.   |
| <span style="border: 1px solid black; border-radius: 5px; padding: 2px;">CTRL/C</span> | Interrupt current routine execution.  |
| <span style="border: 1px solid black; border-radius: 5px; padding: 2px;">CTRL/H</span> | Same as the Backspace character.  |
| <span style="border: 1px solid black; border-radius: 5px; padding: 2px;">CTRL/I</span> | Same as the <span style="border: 1px solid black; border-radius: 5px; padding: 2px;">TAB</span> character.                                    |
| <span style="border: 1px solid black; border-radius: 5px; padding: 2px;">CTRL/L</span> | Causes a form feed operation.   |
| <span style="border: 1px solid black; border-radius: 5px; padding: 2px;">CTRL/M</span> | Same as the <span style="border: 1px solid black; border-radius: 5px; padding: 2px;">RET</span> character.                                    |
| <span style="border: 1px solid black; border-radius: 5px; padding: 2px;">CTRL/O</span> | Suspends output until next <span style="border: 1px solid black; border-radius: 5px; padding: 2px;">CTRL/O</span> is received (data is lost). |
| <span style="border: 1px solid black; border-radius: 5px; padding: 2px;">CTRL/Q</span> | Resume output to terminal suspended by <span style="border: 1px solid black; border-radius: 5px; padding: 2px;">CTRL/S</span> character.      |
| <span style="border: 1px solid black; border-radius: 5px; padding: 2px;">CTRL/R</span> | Reprints the current input line.  |
| <span style="border: 1px solid black; border-radius: 5px; padding: 2px;">CTRL/S</span> | Suspends output to the terminal (no data lost).   |
| <span style="border: 1px solid black; border-radius: 5px; padding: 2px;">CTRL/U</span> | Deletes the current input line.   |
| <span style="border: 1px solid black; border-radius: 5px; padding: 2px;">CTRL/X</span> | Deletes the current input line.   |

## MSM Operators

| Name           | Oper | Sample | Description                              |
|----------------|------|--------|--|
| Addition       | +    | X+Y    | Add Y to X                               |
| And            | &    | X&Y    | Tests if X and Y are both 'true'         |
| Concatenate    | _    | X_Y    | Appends the string Y to the string X     |
| Divide         | /    | X/Y    | Divides X by Y                           |
| Equal To       | =    | X=Y    | Tests if Y is equal to X                 |
| Exponentiation | **   | X**Y   | Raises X to the Y power                  |
| Greater Than   | >    | X>Y    | Tests if X is numerically greater than Y |
| Hexadecimal    | #    | #X     | Converts X from Hexadecimal to Decimal   |
| Indirection    | @    | @X     | Uses the contents of X as the value      |
| Integer Divide | \    | X\Y    | Performs integer division of X by Y      |
| Less Than      | <    | X<Y    | Tests if X is numerically less than Y    |

| <b>Name</b>        | <b>Oper</b> | <b>Sample</b> | <b>Description</b>   |
|--------------------|-------------|---------------|--|
| Minus (Unary)      | -           | -X            | Reverses the arithmetic sign of X                          |
| Modulo             | #           | X#Y           | Performs modulo division of X by Y                         |
| Multiply           | *           | X*Y           | Multiplies X by Y  |
| Not                | '           | 'X            | Reverses truth value of X                                  |
| Or                 | !           | X!Y           | Tests if either X or Y is true                             |
| Pattern Match      | ?           | X?Pat         | Tests if X matches pattern specified by Pat                |
| Plus (Unary)       | +           | +X            | Forces numeric conversion of X                             |
| String Contains    | [           | X[Y           | Tests if string Y is contained in X                        |
| String Follows     | ]           | X]Y           | Tests if string Y collates after X                         |
| String Sorts After | ]]          | X]]Y          | Tests if string Y follows X in subscript ordering sequence |
| Subtraction        | -           | X-Y           | Subtracts Y from X   |

## MSM Pattern Match Characters

| Code     | Description  |
|----------|--|
| A or a   | The 52 alphabetic characters, 'A' through 'Z' and 'a' through 'z'  |
| C or c   | The ASCII control characters (i.e., decimal values 0 through 31 and 127)   |
| E or e   | All 256 characters including 128 ASCII and 128 extended characters   |
| L or l   | The 26 lower-case alphabetic characters, 'a' through 'z'   |
| N or n   | The ten numeric digits, 0 through 9  |
| P or p   | The 33 punctuation characters including the <span style="border: 1px solid black; padding: 0 2px;">SP</span> character |
| U or u   | The 26 upper-case alphabetic characters, 'A' through 'Z'   |
| ( , )    | Starts pattern groups, separates individual patterns, ends pattern groups.   |
| "str"    | Specified characters in exactly the same order.  |
| Int      | The pattern must match the number of times specified by Int  |
| .        | The pattern must match none, one, or more times  |
| Int.Int1 | The pattern must match from Int to Int1 number of times  |

## MSM Commands

**B{REAK}{:Post}** (SP) (SP)

Interrupts program execution to allow debugging.

**B{REAK}{:Post}** (SP) {*Expression*}

Enables BREAK if the value of Expression is 1 and disables it if the value is 0. If the value is 2, then Version 2.0 mode error processing is used. If the value is -2 then Post-Version 2.0 mode is used.

**C{LOSE}{:Post}** (SP) *Dev{:Parm}*{,...}

Releases a device that is owned by the current job.

**D{O}{:Post}** (SP) (SP)

Calls a block structured subroutine (immediately following command line) and continues with next command upon completion of the subroutine.

**D{O}{:Post}** (SP) *EntryRef{:Post}*{,...}

Calls a subroutine and then continues with the next command upon completion of the subroutine.

**D{O}{:Post}** (SP) *EntryRef({PrmLst})*{,...}

Calls a subroutine with a formal parameter list and continues with next command upon completion.

**E{LSE}** (SP) (SP)

Processes commands on the remainder of the line only if the value of \$TEST is false.

**F{OR}** (SP) (SP)

Repeats execution of all commands that follow the FOR until a GOTO or QUIT is encountered.

**F{OR}** SP *Local=ForParm*{,...}

*ForParm*=    *Expression*  
              *Begin:Incr*  
              *Begin:Incr:End*

Repeats execution of commands following the FOR on a line for a specified sequence of values of a local variable.

**G{OTO}**{*:Post*} SP *EntryRef*{*:Post*}{,...}

Transfers control to another statement in a program or to an entirely new program.

**H{ALT}**{*:Post*} SP SP

Terminates the user's session.

**H{ANG}**{*:Post*} SP *Expression*{,...}

Suspends program execution for a specified period of time (*Expression* is in seconds).

**I{F}** SP SP

Executes the remainder of the command line if the value of \$TEST is true.

**I{F}** SP *TruthExp*{,...}

Tests the validity of a truth-valued expression and executes the remainder of the command line if it is true.

**J{OB}**{*:Post*} SP *JobParm*{,...}

Where *JobParm* is

*EntryRef*{*[UCI,Vol]*}{*:PartSiz*}{*:Time*}

Initiates execution of a new MUMPS process, at the specified location with the specified size. Returns job number in \$ZA variable.

**K{ILL}** (SP) (SP)

Removes all local variables.

**K{ILL}{:Post}** (SP) *Variable*{,...}

Removes each of the named local or global variables.

**K{ILL}{:Post}** (SP) (*Local*{,...})

Removes all local variables except those explicitly named.

**L{OCK}** (SP) (SP)

Unlocks all variables previously locked by the LOCK command.

**L{OCK}{:Post}** (SP) {±} *Variable*{:*Time*}{,...}

If + or - is not specified, then unlocks all previously locked variables and locks the specified variable. Otherwise, the specified variable is added (+) to or deleted (-) from the list of locked variables.

**L{OCK}{:Post}** (SP) {±} (*Variable*,...){:*Time*}

If + or - is not specified, then unlocks all previously locked variables and locks the specified variables. Otherwise, the specified variables are added (+) to or deleted (-) from the list of locked variables.

**M{ERGE}{:Post}** (SP) *SrcVar=DestVar*{,...}

Copies data from specified local or global variable (i.e., *SrcVar*) to specified local or global variable (i.e., *DestVar*).

**N{EW}** (SP) (SP)

Stacks all local variables.

**N{EW}{:Post}** (SP) *Local*{,...}

Stacks each of the specified local variables.

**N{EW}{:Post}** SP (*Local*{,...})

Stacks all local variables except those explicitly named.

**O{PEN}{:Post}** SP *Dev*{:{*Parm*}{:*Time*}}{,...}

Obtains ownership of a device and initializes it using the specified parameters.

**Q{UIT}{:Post}** SP SP

Terminates execution of a DO command or FOR command.

**Q{UIT}{:Post}** SP *Expression*

Terminates Extrinsic function and returns a value.

**R{EAD}{:Post}** SP *ReadParm*{,...}

*ReadParm*= *StringLit*  
*Format*  
*Variable*{#*Length*}{:*Time*}

Reads information from the current device.

**S{ET}{:Post}** SP *Variable*=*Expression*{,...}

Assigns a value to the specified local or global variable.

**S{ET}{:Post}** SP (*Variable*{,...})=*Expression*

Assigns a value to each of the specified local or global variables.

**S{ET}{:Post}** SP *\$PIECE*(...)=*Expression*

Assigns a value to the specified piece of the local or global variable.

**TC{OMMIT}{:Post}**

Indicates to the system that the current transaction or sub-transaction is complete.

## **TRE{START}{:Post}**

Indicates to the system that the current transaction should be restarted.

## **TRO{LLBACK}{:Post}**

Indicates to the system that the current transaction was not successful and all updates associated with the transaction should be discarded.

## **TS{TART}{:Post} SP Local{:TrnsPrm}**

Indicates the start of a transaction to the system, the local variables that are to be saved, and keyword parameters for the transaction.

## **U{SE}{:Post} SP Dev{:Prm}{:Space}}{,...}**

Designates a specified device as the current device.

## **V{IEW}{:Post} SP {-}Block{,...}**

Reads a block from disk or writes (if negative) a block to disk.

## **V{IEW}{:Post} SP ViewParm{,...}**

where ViewParm is of the form:

*Address{:Domain}:Exp{:Length}{:Type}}*

Modifies the specified memory location.

## **W{RITE}{:Post} SP WriteParm{,...}**

*WriteParm = Format  
Expression  
\*Integer*

Writes the appropriate format characters, Expression value, or ASCII character corresponding to Integer value to the current device, or performs the control function specified by Integer on the current device.

**X**{ECUTE}{:*Post*} SP *Exp*{:*Post*}{,...}

Interprets and executes the commands specified by the expression.

**ZA**{LLOCATE}{:*Post*} SP *Var*{:*Time*}{,...}

Locks the specified local or global variable without affecting previously locked variables.

**ZA**{LLOCATE}{:*Post*} SP (*Var*{,...}){:*Time*}

Locks each of the specified local or global variables without affecting previously locked variables.

**ZD**{EALLOCATE}{:*Post*}

Unlocks all local and global variables previously locked with the ZALLOCATE command.

**ZD**{EALLOCATE}{:*Post*} SP *Variable*{,...}

Unlocks the specified local or global variable without affecting other previously locked variables.

**ZD**{EALLOCATE}{:*Post*} SP (*Variable*{,...})

Unlocks each of the specified local or global variables without affecting other previously locked variables.

**ZF**{LUSH}{:*Post*} SP SP

Writes all modified buffers in the cache to disk and invalidates all entries in the buffer pool.

**ZG**{O}{:*Post*} SP SP

Resumes execution of a program that has been suspended by a BREAK command.

**ZHOROLOG**{:*Post*} (SP) { { $\pm$ }*Dte*}:{ $\pm$ }*Time* }

Used to temporarily adjust the *Date*, the *Time*, or the *Date* and *Time* that is returned by the \$HOROLOG function.

**ZI**{**NSERT**}{:*Post*} (SP) *Exp*{:*LineRef*}{,...}

Inserts a line of code into the current routine at the specified location or active insert point.

**ZL**{**OAD**}{:*Post*} (SP) {*RoutName*}{,...}

Loads a routine from the database or from the current device.

**ZM**{**SM**}{:*Post*} (SP) *Expression*{:*Dev*}

Enables or disables the trace function for DO, GOTO, XECUTE, and QUIT commands.

**ZN**{**EW**}{:*Post*} (SP) *Local*{,...}

Used to make temporary copies of one or more unsubscripted local variables.

**ZP**{**RINT**}{:*Post*} (SP) {*BLine*:{*ELine*}}{,...}

Writes all or part of a routine to the current device.

**ZQ**{**UIT**}{:*Post*} (SP) (SP)

Returns to the next previous execution level that contains an error handling routine.

**ZR**{**EMOVE**}{:*Post*} (SP) {*BLine*:{*ELine*}}{,...}

Removes one or more lines from the current routine.

**ZS{AVE}{:Post}** SP {RoutName{:Level}}{,...}

| <i>Level</i> | <b>Source Code Retained</b>         |
|--------------|-------------------------------------|
| 0            | All text lines                      |
| 1            | All comment lines (; or ;;)         |
| 2            | First line if comment line          |
| 3            | First line comment and all ;; lines |
| 4            | All comment lines (; or ;;)         |
| 5            | All comment lines (; or ;;)         |
| 6            | All ;; comment lines                |
| 7            | All ;; comment lines                |
| 8            | All ; comment lines                 |
| 9            | All ; comment lines                 |
| 10           | First line if comment line          |
| 11           | First line if comment line          |
| 12           | All ; comment lines                 |
| 13           | All ; comment lines                 |
| 14           | Remove all text                     |
| 15           | Remove all text                     |

Saves current routine in the partition and optionally removes all or part of the source code.

**ZSY{NC}{:Post}**

Used to ensure that all buffered DDP requests to another system have been completed.

**ZT{RAP}{:Post}** SP *String*

Forces an error condition during execution of a routine.

**ZU{SE}{:Post}** SP *Dev*

Temporarily gains ownership of a device.

**ZW{RITE}{:Post}** SP {Variable{,...}}

Writes the value of the specified local variable, and any descendents, or the contents of the entire local symbol table.

## MSM Functions

### **\$EntryRef**(*{ActList}*)

Returns a single value from a user defined extrinsic function.

### **\$A**{SCII}(*String*{*,Position*})

Returns the decimal value of a specified ASCII character.

### **\$C**{HAR}(*Decimal*{*,...*})

Translates a list of Decimal expressions into a string of characters whose ASCII codes are those decimal numbers.

### **\$D**{ATA}(*Variable*)

Returns a value that indicates whether a variable has data, has descendents, has both, or has neither. The following defines the values that are returned by the function.

- 0    The global or local variable is not defined, has no data, and has no descendents.
- 1    The global or local variable is defined, has data, but has no descendents.
- 10   The global or local variable is defined, has no data, and has descendents.
- 11   The global or local variable is defined, has data, and has descendents.

### **\$E**{XTRACT}(*String*{*,Begin*{*,End*}})

Returns a substring from a starting location to an ending location within a specified string.

## **\$F{IND}(String,Substring{,Begin})**

Returns an integer value that denotes the location of the first character that follows the specified Substring within the specified String.

## **\$FN{UMBER}(String,Format{,Fraction})**

Returns the value of an expression edited according to the specified format. Fraction is the number of digits to the right of the decimal point to return. The format characters are defined as:

- + Force '+' on positive numbers
- Suppress '-' on negative numbers
- , Insert commas every third position to the left of the decimal point
- P Represent negative numbers in parentheses
- T Represent number with trailing '+' or '-' sign

## **\$G{ET}(Variable{,Value})**

Returns the data value of a Variable if it exists. Otherwise it returns the specified Value or null if a Value is omitted.

## **\$J{USTIFY}(String,Field{,Fraction})**

Returns the value of an expression, right-justified in spaces within a field of specified size. Fraction is the number of digits to the right of the decimal point to return.

## **\$L{ENGTH}(String{,Substring})**

Returns the length of a string, or the number of pieces delimited by a substring that occur within a string.

**\$NA{ME}(Variable{,Subscripts})**

Returns the full global reference, including the indicated number of subscripts, for the specified global.

**\$N{EXT}(Variable{,Direction})**

Identifies the next or previous subscript, in collating sequence, for a specified subscripted variable.

**\$O{RDER}(Variable{,Direction})**

Identifies the next or previous subscript, in collating sequence, for a specified subscripted variable.

**\$P{IECE}(String,Delimiter{,First{,Last}})**

Returns a substring between two occurrences of a specified delimiter. If the \$PIECE function is on the left side of the equal sign in an expression, then the specified piece is appropriately set.

**\$Q{UERY}(Variable{,Direction})**

Identifies the next or previous full global or local reference, in collating sequence, for a specified subscripted variable.

**\$R{ANDOM}(Integer)**

Returns a pseudo-random number in a specified interval (from 0 to Integer - 1, inclusive).

**\$RE{VERSE}(String)**

Returns a string whose characters are in reverse order of the characters contained in the specified *String*.

**\$S{ELECT}(Condition:Expression{,...})**

Returns the value of the first Expression whose truth-valued Condition is true.

**\$T{EXT}**(*EntryRef*)

Returns the text of the specified line within the current routine or a specified routine.

**\$TR{ANSLATE}**(*String,From{,To}*)

Returns a string that has been translated according to the specified masks.

**\$V{IEW}**(*Address{,Domain{,Len{,Type}}}*)

Returns the text, hexadecimal, decimal, or binary value of a specified memory location. Negative Domain values indicate system memory, 0 indicates the VIEW buffer, and positive values correspond to partitions. Types are: 0 - decimal, 1 - string, 2 - hexadecimal, 3 - binary.

**\$ZBname**(*BitParm{,BitParm{,BitParm}}*)

Performs the bit functions specified by *name* using the parameters specified by *BitParm*.

**\$ZBN**(*Name{,Block}*)

Returns the starting block number for a routine or a global. Or, it allocates or deallocates a disk Block.

**\$ZB{OOLEAN}**(*String1,String2,Mask*)

Returns a value that is the result of a specified Boolean operation applied to two arguments. The mask is defined as follows:

| String1 | String2 | Mask |
|---------|---------|------|
| 0       | 0       | 8    |
| 0       | 1       | 4    |
| 1       | 0       | 2    |
| 1       | 1       | 1    |

## **\$ZCR{C}(String{,Type})**

Returns the ASCII summation or Exclusive OR value of all characters contained in the String. The type is defined as:

- 0 Perform Exclusive OR summation (default)
- 1 Perform ASCII summation

## **\$ZD{ATE}(Integer{,Type})**

Returns the external representation of a date by converting it from its internal \$HOROLOG format. The type is defined as:

- 1 Format is MM/DD/YY (Default Type)
- 2 Format is DD MMM YY
- 3 Format is DD/MM/YY (European)

## **\$ZDE{VICE}(Expression)**

Returns the name of the terminal device specified by the numeric expression.

## **\$ZH{EX}(Expression)**

Returns the decimal value of a hexadecimal number (quoted string) or the hexadecimal value of a decimal number (integer value).

## **\$ZN{EXT}(Variable{,Direction})**

Identifies next or previous reference, in collating sequence, for specified subscripted variable.

## **\$ZO{RDER}(Variable{,Direction})**

Identifies next or previous reference, in collating sequence, for specified subscripted variable.

## **\$ZOS(Expression)**

Returns the result of the Host Operating System command specified by the Expression supplied to the function.

## **\$ZP{REVIOUS}(Variable)**

Identifies the previous subscript, in collating sequence, for a specified subscripted variable.

## **\$ZS{ORT}(Variable{,Direction})**

Identifies the next or previous subscript, in collating sequence, for a specified subscripted variable.

## **\$ZU{CI}(UCIName,{VolName})**

Returns the internal number of the specified User Class Identifier within the specified volume group.

## **\$ZU{CI}(UCINum{,VolNum})**

Returns the external name of the specified User Class Identifier number within the specified volume group number.

## **\$ZV{ERIFY}(Block{,{Count}}{,VolNum}))**

Returns a string of errors, if any, that exist in the logical structure of the database. Errors are returned in the following format:

ErrCode,Block:...Block,Offset

When more than one error is detected, errors are separated by a circumflex (i.e., ^).

## MSM Special Variables

### **\$D{EVICE}**

Contains a value that indicates the success or failure of the last Input/Output operation processed for the current device.

### **\$H{OROLOG}**

Contains the current date and time as integer values separated by a comma.

### **\$I{O}**

Contains the number of the current I/O device.

### **\$J{OB}**

Contains the job number of the job currently executing.

### **\$K{EY}**

Contains the control sequence which terminated the last READ command from the current device.

### **\$P{RINCIPAL}**

Contains the number of the job's Principal device.

### **\$S{TORAGE}**

Contains the decimal value equivalent to the number of bytes remaining in the partition.

### **\$SY{STEM}**

Contains the MUMPS User's Group assigned vendor identification number (i.e., 43) for MSM and the Micronetics assigned serial number of the current MSM license.

## **\$T{EST}**

Contains the truth value of the most recently executed IF command, or timeout.

## **\$TL{EVEL}**

Indicates whether or not a transaction is currently in progress. A value of zero (0) indicates that no transaction is in progress and any other value indicates that a transaction is in progress.

## **\$TR{ESTART}**

Contains a count of the number of transaction restarts that have occurred during the transaction.

## **\$X**

Contains the decimal value of the current output column for the current device.

## **\$Y**

Contains the decimal value of the current output row for the current device.

## **\$ZA**

Contains device specific information for the current device.

## **\$ZB**

Contains device specific information for the current device.

## **\$ZC**

Contains device specific information for the current device.

## **\$ZE{RROR}**

Contains the text of the error message most recently produced by the MSM system.

## **\$ZL{EVEL}**

Contains the number of the current nesting level.

## **\$ZN{AME}**

Contains the name of the routine currently loaded into memory.

## **\$ZO{RDER}**

Returns the data value of the next global node that follows the current global reference.

## **\$ZR{EFERENCE}**

Contains the full global reference, including name and subscripts, of the last global referenced.

## **\$ZT{RAP}**

Contains the line label and the routine name of the program that is to receive control when an error occurs.

## **\$ZV{ERSION}**

Contains the name and release number of the MSM software being used.

## **MSM Structured System Variables**

### **^\$DEVICE**

Contains information about devices attached to the system.

### **^\$GLOBAL**

Contains information about globals stored in the current UCI.

### **^\$JOB**

Contains information about jobs that are currently executing.

### **^\$LOCK**

Contains LOCKS that have been acquired by jobs.

### **^\$ROUTINE**

Contains information about routines stored in the current UCI.

## MSM Device Numbers

| Number  | Description   |
|---------|---|
| 1       | Primary Console Device  |
| 2       | Spooling/Despooling Device  |
| 3-19    | Terminal Devices (Serial/Parallel)                                |
| 20-46   | Routine Interlock Devices   |
| 47-50   | Magnetic Tape Devices   |
| 51-54   | Host File Server Units  |
| 55      | Host System Spooling Device                                       |
| 56-58   | Reserved for Specialized Drivers                                  |
| 59-62   | Sequential Block Processors                                       |
| 63      | VIEW Device for Disk and Memory                                   |
| 64-199  | Terminal Devices (Serial/Parallel)                                |
| 200-223 | Routine Interlock Devices   |
| 224-255 | In-Memory Job Communication<br>Even = Receiver, Odd = Transmitter |
| 256+    | Terminal Devices (Serial/Parallel)                                |

## MSM Device Parameters

### TERMINAL DEVICES

OPEN{:Post} [SP] Dev{: (opt1:....:opt13)}{:Time}

USE{:Post} [SP] Dev{: (opt1:....:opt13)}{:Space}

CLOSE{:Post} [SP] Dev

- opt1 The right margin (0 through 255) for the device. When \$X is equal to the right margin, the system will insert a carriage return/line feed sequence.
- opt2 Reserved for Future Use.
- opt3 Maximum number of characters (0 through 2044) to accept on input before terminating the read.
- opt4 Reserved for Future Use.
- opt5 One or more device characteristics to be set for the terminal. Each of the device characteristics is described in the table below.
- opt6 One or more device characteristics to be cleared for the terminal. Each of the device characteristics is described in the table below.
- opt7 The value to be assigned to the \$X and the \$Y Special Variables. It is passed as a number in the form \$Y\*256+\$X.
- opt8 This parameter is used to set the baud rate, parity, data bits, stop bits, etc. for the device. This parameter is machine specific in MSM.
- opt9 String of \$ASCII control characters (0 to 31 and 127) that are to be recognized as READ terminators. Default is \$C(13,27).

- opt10 This parameter specifies the name of the device connected to the port.
- opt11 This parameter specifies the size of the text buffer to be obtained for a window device (MSM-PC only).
- opt12 This parameter specifies the maximum size of the window on the device. The size is specified as a single number that is equal to the number of rows times 256 plus the number of columns (MSM-PC only).
- opt13 This parameter specifies the actual location of the window on the device. The location is specified as a single number that is equal to the row number times 256 plus the column number of the upper left corner of the window (MSM-PC only).

| Bit | Description of Device Characteristic         |
|-----|--|
| 31  | Reserved for future use                      |
| 30  | Reserved for future use                      |
| 29  | Reserved for future use                      |
| 28  | Reserved for future use                      |
| 27  | If set, honor XON/XOFF when bit 23 is set    |
| 26  | If set, no typeahead flush on prompted reads |
| 25  | If set, typeahead is disabled                |
| 24  | If set, disable ZUSE for this device         |
| 23  | If set, don't interpret control characters   |
| 22  | If set, don't generate XOFF on buffer full   |
| 21  | If set, CTRL/O is treated as normal data     |
| 20  | If set, ignore non-MSM control characters    |
| 19  | If set, read terminated on empty line delete |
| 18  | If set, allow input of full 8 bit characters |
| 17  | If set, WRITE *n doesn't change \$X or \$Y   |
| 16  | If set, ANSI type terminal, otherwise VT52   |
| 15  | If set, interrupt was received from terminal |
| 14  | If set, lower-case converted to upper-case   |
| 13  | If set, no line feed after carriage return   |

| Bit | Description of Device Characteristic          |
|-----|---|
| 12  | If set, TAB is not expanded to spaces         |
| 11  | If set, line carrier signal is present        |
| 10  | If set, indicates device is printer           |
| 9   | If set, DTR is asserted on port               |
| 8   | If set, device is not permitted to logon      |
| 7   | If set, position cursor on \$X and \$Y update |
| 6   | If set, ESCAPE processing feature is enabled  |
| 5   | If set, CTRL/S was received from terminal     |
| 4   | If set, CTRL/O was received from terminal     |
| 3   | If set, line is modem controlled              |
| 2   | If set, CRT and erase using backspace         |
| 1   | If set, terminal is output-only device        |
| 0   | If set, characters received are not echoed    |

**\$ZA** A 32-bit value that corresponds to the device characteristics described above.

**\$ZB** A value that indicates how the input command was terminated.

13 Carriage return

27 Escape

nn user specified terminator (nn)

0 fixed length reached

If Escape Processing is enabled  
(Escape value\*256)+27

**\$ZC** Contains 0 (zero) if the operation completed normally or 1 if a device error occurred.

# SEQUENTIAL BLOCK PROCESSOR

OPEN{:Post} [SP] Dev{:(opt1:....:opt5)}{:Time}

USE{:Post} [SP] Dev{:(opt1:....:opt5)}{:Space}

CLOSE{:Post} [SP] Dev

- opt1 The block offset indicating the starting byte within the block where the SBP is to be positioned. The value can be from 0 through 1011.
- opt2 The disk block number indicating where the SBP is to be positioned within the database. The block number is relative to the beginning of the database.
- opt3 The database index number which indicates which of the MSM databases contains the SBP area.
- opt4 A single character ASCII value that is used to separate records in stream format.
- opt5 The format to be used to read or write from the file. This can be either 'S' for stream format or 'V' for variable length format.
- \$ZA Relative block number of the disk buffer currently being processed.
- \$ZB Offset in the block to the next byte to be processed.
- \$ZC Contains 0 (zero) if the operation completed normally, -1 if end-of-file was reached, or 1 if a device error occurred.

## HOST FILE SERVER

OPEN{:Post} [SP] Dev{:(opt1:....:opt6)}{:Time}

USE{:Post} [SP] Dev{:(opt1:....:opt6)}{:Space}

CLOSE{:Post} [SP] Dev{:(opt1)}

- opt1 The full name (i.e., path, file, extensions, etc.) of the file to be accessed (e.g., C:\MSM\DATA or /usr/msm/testdata or [USER]DATA.DAT etc.).
- opt2 The access mode indicating how the file will be used. Valid modes are 'R' for READ, 'W' for WRITE, 'M' for Mixed (READ and WRITE), 'B' for Buffer (share View buffer), 'C' for Control (block read and write), 'N' for No Close, and 'A' to append to end of file. The default is 'R'.
- opt3 The file offset indicating the starting byte within the file where the HFS is to be positioned. The value is relative to the location specified by option 4. A signed value (i.e., -5) can be specified to move forward or backward from the current position in the file.
- opt4 In block or control mode, this contains the relative block number within the file. In any other mode, this specifies the type of offset in option 3. A value of 0 indicates offsets from the beginning of the file, 1 for offsets from the current position, and 2 for offsets from the end of the file. The default for this option is 0.
- opt5 Specifies the data recording format. Valid formats are 'S' for stream and 'V' for variable length.
- opt6 READ Terminator (1 to 3 ASCII characters or null) is used to specify the sequence that separate records in stream format. The default value is \$C(10) except on MSM-PC where it is \$C(13,10).

- \$ZA Contains the number of bytes transferred to or from the file. This can be used to test for an end-of-file condition by comparing it to the number of bytes that should have been transferred. If they do not agree, then the end of file has been reached. If an error occurs on the transfer, this field will contain a -1. In addition, immediately after OPEN command, \$ZA will be 0 if the file was successfully opened. If not, \$ZA will be -1.
- \$ZB Contains the current block number if the file was opened in buffer mode or control mode. In any other mode, this field will contain the current byte offset into the file.
- \$ZC Contains 0 (zero) if the operation completed normally, -1 if end-of-file was reached, or 1 if a device error occurred.

## MAGNETIC TAPE DEVICES

OPEN{:Post} SP Dev{: (opt1:....:opt5)}{:Time}

USE{:Post} SP Dev{: (opt1:....:opt5)}{:Space}

CLOSE{:Post} SP Dev

opt1 Indicates the mode for tape access.

- A ASCII Character Set is to be used.
- B Buffer transfers using the VIEW buffer. Only modes T, C, 3, 4, 5 and WRITE \* commands are allowed.
- C Continuous mode (double buffering).
- E EBCDIC Character Set is to be used.
- F Fixed Length Records for input and stream mode for output. An opt2 value is required and opt3 may be specified.
- L Standard Labels (ASCII labels in A mode and IBM labels in E mode).
- N Indicates the null character ('00') as part of the data in stream mode.
- R READ indicates only input type commands are allowed for the tape.
- S Stream Format for data on output with automatic blocking of information.
- T Tape Mark errors are inhibited.
- U Unlabeled (disables label processing).
- V Variable Length Records (ASCII type in A mode and EBCDIC type in E mode).
- 3 800 BPI density is to be used.
- 4 1600 BPI density is to be used.
- 5 6250 BPI density is to be used.

- opt2 Logical Record Length (1 to 32767 bytes) specifies the size of each record.
- opt3 Physical Block Size (14 to 32767 bytes) of each block Read or Written to tape.
- opt4 Delimiter String (one to three ASCII characters) used to separate logical records in stream mode.
- opt5 Buffer Offset within the block where the next READ or WRITE is to occur.

The following is a list of the WRITE \*N codes that can be used to position the tape.

| Code | Description |
|------|-------------|
|------|-------------|

- |    |   |
|----|---|
| 1  | Backspace one physical block and reset the READ/WRITE switch.         |
| 2  | Forward Space one physical block and reset the READ/WRITE switch.     |
| 3  | Write Tape Mark after flushing buffer if last operation was output.   |
| 4  | Write Block and if tape is at beginning perform label processing.     |
| 5  | Rewind Tape after flushing buffer if last operation was WRITE.        |
| 6  | Read Block and if tape is at beginning perform label processing.      |
| 7  | Read Label if tape is positioned at label, otherwise read next block. |
| 8  | Write Header Labels if tape was opened with label processing.         |
| 9  | Write EOF Label in ASCII or EBCDIC as appropriate.                    |
| 10 | Contents of \$ZA updated with current tape status.                    |
| 11 | Backspace to tape mark and reset.                                     |

- 12 Forward Space tape to next tape mark and reset READ/WRITE switch.
- 13 Erase Tape and reset READ/WRITE switch.
- 14 Retension the Cartridge Tape and reset READ/WRITE switch.
- 15 Forward space tape to end of last data written and reset READ/WRITE switch.
- 16 Rewind and unload Tape after flushing buffer if last operation was WRITE.
- \$ZA Contains status about the last I/O operation to tape. The value is updated only when an error occurs or a block mode status command is executed.
  - 0 Beginning of tape
  - 1 End of tape (physical)
  - 2 End of recorded data
  - 3 Tape positioning in progress
  - 4 Tape unit is on-line
  - 5 Tape unit is ready
  - 6 Tape media has been changed
  - 7 Tape mark detected
  - 8 Logical tape error
  - 9 Block length error
  - 10 Tape media error
  - 11 Tape is write protected
  - 12 Bus error
  - 13 Data overrun
  - 14 Hardware error
  - 15 Reserved for future use
- \$ZB Contains a value that indicates the number of physical bytes transferred on the last READ or WRITE operation. If the operation was successful, this should be the same as the block size specified by opt2 on the OPEN command.
- \$ZC Contains 0 (zero) if the operation completed normally, -1 if end-of-file was reached, or 1 if a device error occurred.

# SPOOL DEVICE

OPEN{:Post} [SP] Dev{:opt} {:Time}

USE{:Post} [SP] Dev{:opt} {:Space}

CLOSE{:Post} [SP] Dev{:opt}

opt     File Designator indicates what Destination Number and what File Number is to be accessed.

For OPEN commands, the value of this option can be in any one of the following forms:

- |          |  |
|----------|--|
| null     | indicates that the file is to be created with the default destination code of 0.                 |
| Dest     | A number from 1 through 255 that specifies the group (i.e., Destination) for the file.           |
| 256+File | opens the file specified by the File number independent of its destination code.                 |
| 512+Dest | opens the first file within the group of files specified by the destination number.              |
| 512      | opens the first file in spool space regardless of the destination code associated with the file. |

On the USE command, the value of this option can only be in the following format:

- |      |   |
|------|---|
| File | makes the file specified by the File number the current file on the spool device. |
|------|---|

On the CLOSE command, the value of this option can be in any one of the following formats:

- |          |  |
|----------|--|
| null     | the spool file associated with the current spool device is closed but not deleted. |
| File     | the file specified by the File number is closed but it is not deleted.             |
| 256+File | the file specified by the File number is closed and it is deleted.                 |
| 256      | the current spool file is closed and it is deleted.                                |

**\$ZA** Contains the destination number and file number of the current spool file in the form Dest-Num\*256+FileNum if the last operation was successful. If not, \$ZA contains an error indicator as one of the following:

- 1 an end-of-file has occurred on the current operation.
- 2 the specified spool file does not exist.
- 3 on OPEN, the file table was full and the new file could not be added.
- 4 an error exists in the logical structure of the spool file.
- 5 the spool space is full and the record could not be added.
- 6 spooling is not enabled.

**\$ZB** Contains the number of the physical disk block being processed by the spool device.

**\$ZC** Contains 0 (zero) if the operation completed normally, -1 if end-of-file was reached, or 1 if a device error occurred.

## HOST SYSTEM SPOOL DEVICE

OPEN{:*Post*} SP *Dev*{:*opt*}{:*Time*}

USE{:*Post*} SP *Dev*

CLOSE{:*Post*} SP *Dev*{:*opt*}

**opt**     Operating System specific parameters to be passed to the Host System Spooler as part of the command that invokes it. These parameters are passed to the spooler as specified on the command. No syntax checking is done by MSM on the parameters.

**\$ZA**    Contains a value of zero if the OPEN command was successful. Otherwise it contains a value of minus one (-1) to indicate that the OPEN failed.

**\$ZB**    Always contains a value of zero. The Special Variable is not used for Host System Spooling.

**\$ZC**    Contains 0 (zero) if the operation completed normally or 1 if an error occurred.











## MSM Full Screen Editor

| <b>Cursor Movement Commands</b> |  |
|---------------------------------|--|
| Backspace                       | Moves cursor one character left        |
| H                               | Moves cursor one character left        |
| J                               | Moves cursor down one line             |
| K                               | Moves cursor up one line               |
| L                               | Moves cursor one character right       |
| space                           | Moves cursor one character right       |
| 0 (zero)                        | Moves cursor to beginning of line      |
| \$                              | Moves cursor to end of current line    |
| <span>RET</span>                | Moves cursor to beginning of next line |
| tab                             | Move cursor ten (10) characters right  |
| Fx                              | Move cursor to next occurrence of x    |
| nG                              | Go to relative line number             |
| <b>Scroll Commands</b>          |  |
| <span>CTRL/E</span>             | Scroll one line forward in program     |
| <span>CTRL/D</span>             | Scroll half screen forward in program  |
| <span>CTRL/F</span>             | Scroll full screen forward in program  |
| <span>CTRL/Y</span>             | Scroll one line back in program        |
| <span>CTRL/U</span>             | Scroll half screen back in program     |
| <span>CTRL/B</span>             | Scroll full screen back in program     |
| <span>CTRL/G</span>             | Display current position in routine    |
| <span>CTRL/L</span>             | Re-display current screen              |
| <span>CTRL/R</span>             | Re-display routine from first line     |
| <b>Delete Commands</b>          |  |
| DE                              | Delete from cursor to end of line      |
| D\$                             | Delete from cursor to end of line      |
| DB                              | Delete from start of line to cursor    |
| D0                              | Delete from start of line to cursor    |
| DL                              | Delete line containing cursor          |
| nDL                             | Delete n lines starting at cursor line |
| DM                              | Delete marked characters/lines         |

| <b>Search Commands</b>        |  |
|-------------------------------|--|
| .S                            | Search for a string starting at cursor |
| .J                            | Search for a label starting at cursor  |
| N                             | Search forward with same string        |
| -N                            | Search backward with same string       |
| <b>Mark Commands</b>          |  |
| M                             | Mark line from cursor to end           |
| B                             | Set ending bound of a marked area      |
| -M                            | Unmark the current marked area         |
| <b>Transfer/Copy Commands</b> |  |
| TM                            | Move marked area after cursor          |
| TC                            | Copy marked area after cursor          |
| P                             | Insert last Yanked or Deleted lines    |
| nY                            | Yank n lines starting at current line  |
| <b>Editing Commands</b>       |  |
| X                             | Delete character at cursor position    |
| nX                            | Delete n characters at cursor position |
| I                             | Insert characters before cursor        |
| A                             | Insert characters after cursor         |
| V                             | Overlaying characters at cursor        |
| O                             | Open new line after current line       |
| -O                            | Open new line before current line      |
| R                             | Replace characters at cursor           |
| U                             | Undo all changes to line               |
| S                             | Split current line after cursor        |
| C                             | Combine current line with next line    |
| <b>Miscellaneous Commands</b> |  |
| .Q                            | Quit the current routine edit          |
| .F                            | File the routine and exit the routine  |
| .W                            | File the routine and continue editing  |
| .E                            | Temporarily edit another program       |
| .R                            | Read in lines from another program     |
| .C                            | Change every occurrence of a string    |
| .P                            | Put lines into temporary storage       |
| .G                            | Get lines from temporary storage       |
| ?                             | Display help information               |

## MSM CUA-Style Full Screen Editor

### Cursor Movement and Scrolling Keys

|   |                                  |
|---|----------------------------------|
|        | Moves cursor one character left  |
|        | Moves cursor one character right |
|        | Moves cursor up one line         |
|        | Moves cursor down one line       |
| CTRL/  | Moves cursor one word left       |
| CTRL/  | Moves cursor one word right      |
| CTRL/  | Moves cursor backward one line   |
| CTRL/  | Moves cursor backward one line   |
| ALT/   | Moves cursor up five lines       |
| ALT/   | Moves cursor down five lines     |
| HOME  | Moves cursor to beginning line   |
| END   | Moves cursor to end of line      |
| PGUP  | Scrolls backward one page        |
| CTRL/PGUP   | Scrolls backward one half page   |
| PGDN  | Scrolls forward one page         |
| CTRL/PGDN   | Scrolls forward one half page    |
| CTRL/HOME   | Scrolls to start of routine      |
| CTRL/END  | Scrolls to end of routine        |
| ALT/HOME  | Scrolls to top of screen         |
| ALT/END   | Scrolls to bottom of screen      |

### Function Key Shortcuts

|    |                                    |
|----|------------------------------------|
| F1 | Help information on current field  |
| F2 | Copy marked text to Paste buffer   |
| F3 | Cut text and store in Paste buffer |
| F4 | Paste buffer to current location   |
| F5 | Undo the last change               |
| F6 | Change specified string            |
| F7 | Find next occurrence of string     |

## Function Key Shortcuts

|          |                 |
|----------|-----------------|
| F8       | Find next       |
| F9       | Select text     |
| F10      | Menu bar        |
| F11      | Accept          |
| F12      | Cancel (escape) |
| SHIFT/F8 | Find previous   |

## Control Key Shortcuts

|        |                        |
|--------|------------------------|
| CTRL/A | Insert above line      |
| CTRL/B | Insert below line      |
| CTRL/C | Close current window   |
| CTRL/D | DOS Shell              |
| CTRL/E | MSM Programmer Shell   |
| CTRL/F | Find                   |
| CTRL/G | Find next              |
| CTRL/J | Jump to label+offset   |
| CTRL/K | Mark current/next word |
| CTRL/L | Mark current line      |
| CTRL/N | Next window            |
| CTRL/O | Open new window        |
| CTRL/P | Print routine          |
| CTRL/R | Retrieve routine       |
| CTRL/S | Save routine           |
| CTRL/T | Syntax check           |
| CTRL/U | Undo line changes      |
| CTRL/W | Same as CTRL/S         |
| CTRL/V | View open windows list |
| CTRL/X | Exit, save routine     |
| CTRL/Y | Highlight all (toggle) |

## MSM Error Messages

An error message, in the following format, is displayed or returned in the \$ZERROR special variable when MSM detects an error.

*<Error> Line^Rtn:Cmd:Arg:Major:Minor:Info*

|              |                                    |
|--------------|------------------------------------|
| <i>Line</i>  | Relative line number               |
| <i>Rtn</i>   | Routine Name                       |
| <i>Cmd</i>   | Command number on the line         |
| <i>Arg</i>   | Argument number on the command     |
| <i>Major</i> | Major number of error              |
| <i>Minor</i> | Minor number of error              |
| <i>Info</i>  | Additional information about error |

### ASync

Indicates that an error has occurred during the asynchronous processing of a previous SET for KILL operation on a Remote Volume Group.

### BADCH

Invalid Kanji or Shift-Jis character encountered.

### BKERR

Interpreter encountered a BREAK command.

### CLOBR

An attempt has been made to overlay the current routine using ZLOAD, ZREMOVE, or ZINSERT or ZSAVE of an active routine.

## **CMMND**

The interpreter encountered an illegal command.

## **DDPER**

The DDP server has encountered an error while trying to process a transaction through the network.

## **DIVER**

An attempt has been made to divide by zero.

## **DKFUL**

Indicates that no more space is available on the disk (within the current UCI limits).

## **DKHER**

Indicates an unrecoverable error while reading from or writing to disk.

## **DKRES**

Indicates an attempt was made to allocate a block in the reserved area of the end of the user's expansion area because the expansion area is almost full.

## **DKSER**

Indicates a block read from disk was not the expected type.

## **DPARM**

Indicates a formal list has been encountered during processing, however, it is not the result of a DO command. Or, an Actual List has been used incorrectly or an invalid use of the QUIT command.

## **DSCON**

Indicates that the terminal connection has been disconnected (e.g., telephone hangup, terminal power off, etc.).

## **DSTDB**

Indicates that an error in transmission has occurred on the network.

## **EXPER**

Indicates an exponentiation error has been detected.

## **FUNCT**

The interpreter has encountered an undefined or improperly used function.

## **INDER**

The interpreter has encountered illegal or incorrect use of the indirection operator.

## **INHIB**

Indicates that access to a database through the network has failed because database reads or writes are inhibited on the specified machine.

## **INRPT**

The system has received an interrupt (i.e., CTRL/C) while BREAK processing was enabled.

## **ISYNT**

An attempt has been made to insert an illegal line of code into the routine buffer.

## **LCNSE**

Indicates the number of jobs exceeds the license limit.

## **LINER**

A reference has been made to a line which does not exist within the current or external routine.

## **MAPER**

A disk block being freed is already marked free in the corresponding MAP block.

## **MERGE**

Indicates a MERGE command has been issued in which the destination operand is a descendent of the source operand.

## **MINUS**

The interpreter encountered a negative number or zero when a positive number was expected.

## **MODER**

An attempt has been made to access the HFS, SBP, or Tape in a mode not consistent with the open parameters.

## **MSMCX**

Indicates the in-memory communications path between MSM tasks has been interrupted.

## **MTERR**

Indicates that the last input or output operation to a tape device resulted in an error condition. This error, which can be analyzed by examining the \$ZA special variable, only occurs if an error handling routine has been specified through \$ZTRAP.

## **MXMEM**

A memory address supplied to the VIEW command or \$VIEW function is outside the limits allowed by MSM.

## **MXNUM**

The value of a number is greater than the largest number allowed.

## **MXSTR**

The value of a string exceeds the maximum length allowed by the system.

## **NAKED**

Access to a global variable using the naked indicator is invalid.

## **NODEV**

An attempt to OPEN a device which has not been defined to the system.

## **NOJRN**

An attempt was made to journal a global in a single-user mode.

## **NOPEN**

An attempt to USE a device which has not been previously OPENED.

## **NOPGM**

A reference has been made to a program that does not exist.

## **NOSYS**

The volume group name specified in an extended global reference does not exist or is not accessible through the network.

## **NOUCI**

The User Class Identifier (UCI) name specified in an extended global reference does not exist or is not accessible through the network.

## **PCERR**

Either a post conditional argument is not allowed on the command or the post conditional argument is invalid.

## **PGMOV**

Insufficient memory space is left in the partition to complete the operation.

## **PLDER**

The compiled code for the specified routine is from an older version of MSM and can not be executed by the current version.

## **PROT**

An attempt to access a global that the user is not authorized to access.

## **SBSCR**

The subscript used in a local or global reference is invalid (i.e., is null or contains \$C(0) character).

## **STKOV**

The system stack has overflowed as a result of nested indirection, program loop, etc.

## **SYNTAX**

The interpreter has encountered a syntax error in the line being executed.

## **SYSTEM**

The MSM system has encountered an internal error. The system should be shutdown as quickly as possible and re-booted.

## **TPROC**

A transaction processing error has occurred.

## **UNDEF**

An attempt to reference a non-existent local or global variable.

## **VWERR**

Indicates that an attempt has been made to access a device in a mode that requires use of the VIEW buffer when the VIEW buffer has not been opened. This also occurs if the VIEW buffer is closed before the device that is sharing it has been closed.

## **ZCALL**

Indicates the function name specified on a \$ZCALL does not exist or the parameters specified are invalid.

## **ZCERR**

Indicates that an error has occurred in the routine that was invoked as a result of a \$ZCALL function.

## **ZSAVE**

Indicates that one of the lines in the routine being compiled is too large to fit in the disk buffer. The line should be broken into two or more lines to correct the problem.

## **ZSVGP**

a ZSAVE command was issued, but the volume group was not enabled for any p-code type.

## MSM Error Numbers

| Major | Minor | Explanation                    |
|-------|-------|--------------------------------|
| 1     | -     | Command type errors            |
|       | 1     | Unrecognized command           |
| 2     | -     | Argument type errors           |
|       | 1     | Missing parenthesis            |
|       | 2     | Missing or bad colon           |
|       | 3     | Missing or bad equals          |
|       | 4     | Missing or bad local variable  |
|       | 5     | Missing or bad global variable |
|       | 6     | Missing or bad function        |
|       | 7     | Missing or bad routine name    |
|       | 8     | Missing or bad routine label   |
|       | 9     | Missing or bad routine offset  |
|       | 10    | Indirect argument error        |
|       | 11    | Argument condition error       |
|       | 12    | Bad argument delimiter         |
|       | 13    | Bad command                    |
| 3     | -     | Expression type errors         |
|       | 0     | Bad special variable name      |
|       | 1     | Bad system function            |
|       | 2     | Bad local variable             |
|       | 3     | Bad global variable            |
|       | 4     | Bad string constant            |
|       | 5     | Bad numeric constant           |
|       | 6     | Unbalanced parenthesis         |
|       | 7     | Invalid syntax in term         |
|       | 8     | Bad operator                   |
|       | 9     | Bad delimiter                  |

| Major | Minor | Explanation                     |
|-------|-------|---------------------------------|
| 4     | -     | Reference type errors           |
|       | 0     | Undefined local variable        |
|       | 1     | Undefined global variable       |
|       | 2     | Undefined label                 |
|       | 3     | Undefined routine               |
|       | 4     | Invalid naked reference         |
|       | 5     | Non-existent device             |
|       | 6     | Unsubscripted local required    |
|       | 7     | Variable reference required     |
|       | 8     | ZLOAD during execution          |
|       | 9     | Undefined UCI reference         |
|       | 10    | Attempt insert of invalid line  |
|       | 11    | Unknown data type               |
|       | 12    | Missing function parameter      |
|       | 13    | Undefined system in reference   |
|       | 14    | Global protection violation     |
|       | 15    | VIEW Protection violation       |
|       | 16    | ZCALL Error                     |
|       | 17    | Formal List without DO parms    |
|       | 18    | QUIT value inside FOR           |
|       | 19    | QUIT value not in extrinsic     |
|       | 20    | QUIT no value in extrinsic      |
|       | 21    | End extrinsic implied QUIT      |
|       | 22    | No formal list, call with parms |
|       | 23    | Actual bigger than formal list  |
|       | 24    | Formal list not single lvn      |
|       | 25    | Duplicate formal list variable  |
|       | 26    | Pass by reference not allowed   |
|       | 27    | MERGE trgt descendent of src    |
|       | 28    | Nested namespace request        |

| Major | Minor | Explanation                     |
|-------|-------|---------------------------------|
| 5     | -     | Value type errors               |
|       | 0     | String exceeded max length      |
|       | 1     | Select function error           |
|       | 2     | Divide by zero                  |
|       | 3     | Negative number                 |
|       | 4     | Maximum number                  |
|       | 5     | Device not open                 |
|       | 6     | Maximum memory                  |
|       | 7     | String value required           |
|       | 8     | Indirection gave null value     |
|       | 9     | Indirection is not a name       |
|       | 10    | Partition not active (\$VIEW)   |
|       | 11    | Invalid VIEW or \$VIEW          |
|       | 12    | Function parm out of range      |
|       | 13    | Invalid subscript               |
|       | 14    | Device access not allowed       |
|       | 15    | Invalid Kanji or Shift-Jis char |
|       | 16    | Not allowed to write block 0    |
|       | 17    | Invalid shared View buffer use  |
|       | 18    | Zero to non-positive power      |
|       | 19    | Neg # to non-integer power      |

| Major | Minor | Explanation                    |
|-------|-------|--------------------------------|
| 6     | -     | Environmental errors           |
|       | 0     | Break key depressed            |
|       | 1     | Unable to expand memory        |
|       | 2     | HALT command executed          |
|       | 3     | LOCK Table full                |
|       | 4     | BREAK command executed         |
|       | 5     | Expression stack overflow      |
|       | 6     | System stack overflow          |
|       | 7     | Old Pcode must be re-saved     |
|       | 8     | DDP Error                      |
|       | 9     | DDP Server Error               |
|       | 10    | DDP database I/O locked        |
|       | 11    | MSM to mumsm failure           |
|       | 12    | Terminal line disconnected     |
|       | 13    | Magnetic tape error            |
|       | 14    | Compiled code too large        |
|       | 15    | ZQuit error                    |
|       | 16    | DDP circuit disabled           |
|       | 17    | ZTRAP command issued           |
|       | 18    | READ exp stack overflow        |
|       | 19    | No p-code for Volume Group     |
|       | 20    | P-code type not available      |
|       | 21    | Users exceed license limit     |
|       | 22    | Asynchronous error on RVG      |
|       | 23    | Journal request, single-user   |
|       | 24    | Transaction processing error   |
| 7     | -     | Disk type errors               |
|       | 0-32  | Block type mismatch            |
|       | 33    | Hardware disk I/O error        |
|       | 34    | Disk full condition            |
|       | 35    | Block number mismatch          |
|       | 36    | Key/Data exceeds max length    |
|       | 37    | Can't open requested database  |
|       | 38    | Block being freed is free      |
|       | 39    | Invalid block number           |
|       | 40    | Block allocation reserved area |

## ANSI 3.64-1979 Mnemonic Namespace

| Mnemonic                                      | Function   |
|---|--|
| /BEL  | Ring Bell  |
| /BS   | Backspace Cursor   |
| /CBT( <u><i>n</i></u> )                       | Backup <u><i>n</i></u> Tab Stops   |
| /CHA( <u><i>n</i></u> )                       | Move Cursor to Column <u><i>n</i></u>  |
| /CHT( <u><i>n</i></u> )                       | Forward <u><i>n</i></u> Tab Stops  |
| /CNL( <u><i>n</i></u> )                       | Cursor Down <u><i>n</i></u> Lines  |
| /CPL( <u><i>n</i></u> )                       | Cursor Up <u><i>n</i></u> Lines  |
| /CR   | Column 1 of Current Line   |
| /CTC( <u><i>a</i></u> , <u><i>b</i></u> ,...) | Set Tab Stops  |
| /CUB( <u><i>n</i></u> )                       | Cursor Back <u><i>n</i></u> Columns  |
| /CUD( <u><i>n</i></u> )                       | Cursor Down <u><i>n</i></u> Rows   |
| /CUD( <u><i>r</i></u> , <u><i>c</i></u> )     | Cursor To Row <u><i>r</i></u> , Column <u><i>c</i></u>   |
| /CUU( <u><i>n</i></u> )                       | Cursor Up <u><i>n</i></u> Lines  |
| /DA( <u><i>a</i></u> , <u><i>b</i></u> ,...)  | Set Device Attributes  |
| /DAQ( <u><i>a</i></u> , <u><i>b</i></u> ,...) | Define Area Qualifications   |
| /DCH( <u><i>n</i></u> )                       | Delete <u><i>n</i></u> Characters at Cursor  |
| /DL( <u><i>n</i></u> )                        | Delete <u><i>n</i></u> Lines at Cursor   |
| /DSR( <u><i>a</i></u> , <u><i>b</i></u> ,...) | Device Status Request  |
| /ECH( <u><i>n</i></u> )                       | Erase <u><i>n</i></u> Characters at Cursor   |
| /ED( <u><i>n</i></u> )                        | Erase Display<br><u><i>n</i></u> =0: Cursor to End of Display<br><u><i>n</i></u> =1: Home to Cursor<br><u><i>n</i></u> =2: Entire Display                  |
| /EL( <u><i>n</i></u> )                        | Erase <u><i>n</i></u> Lines<br><u><i>n</i></u> =0: Cursor to End of Line<br><u><i>n</i></u> =1: Start of Line to Cursor<br><u><i>n</i></u> =2: Entire Line |
| /FF   | Cursor Down 1 Row  |
| /HPA( <u><i>n</i></u> )                       | Cursor to Column <u><i>n</i></u>   |
| /HPR( <u><i>n</i></u> )                       | Cursor Forward <u><i>n</i></u> Columns   |
| /HT   | Horizontal Tab   |
| /HTS  | Set Horizontal Tab Stop  |
| /HVP( <u><i>r</i></u> , <u><i>c</i></u> )     | Cursor to Row <u><i>r</i></u> and Column <u><i>c</i></u>   |

| Mnemonic  | Function                                  |
|---|---|
| /ICH( <u><i>n</i></u> )                         | Insert <u><i>n</i></u> Blanks at Cursor   |
| /IL( <u><i>n</i></u> )                          | Insert <u><i>n</i></u> Lines at Cursor    |
| /IND  | Move Cursor 1 Column Down                 |
| /LF   | Move Cursor 1 Line Down                   |
| /MSMRC( <u><i>n</i></u> )                       | Restore Saved Cursor Position             |
| /MSMRGN( <u><i>n</i></u> , <u><i>m</i></u> )    | Setup Scrolling Region                    |
| /MSMRM( <u><i>a</i></u> , <u><i>b</i></u> ,...) | Reset Modes                               |
| /MSMSC  | Save Current Cursor Position              |
| /MSMSM( <u><i>a</i></u> , <u><i>b</i></u> ,...) | Set Modes                                 |
| /NEL  | Move to Column 1 Next Line                |
| /NP( <u><i>n</i></u> )                          | Advance <u><i>n</i></u> Pages of Terminal |
| /PP( <u><i>n</i></u> )                          | Backup <u><i>n</i></u> Pages of Terminal  |
| /RI   | Reverse Index (Up 1 Column)               |
| /RIS  | Reset to Initial State                    |
| /RM( <u><i>a</i></u> , <u><i>b</i></u> ,...)    | Reset Modes                               |
| /SGR( <u><i>a</i></u> , <u><i>b</i></u> ,...)   | Set Graphical Rendition                   |
| /SM( <u><i>a</i></u> , <u><i>b</i></u> ,...)    | Set Modes                                 |
| /TBC( <u><i>a</i></u> , <u><i>b</i></u> ,...)   | Clear Tab Stops                           |
| /VPA( <u><i>r</i></u> )                         | Move to Row <u><i>r</i></u> Same Column   |
| /VPA( <u><i>n</i></u> )                         | Cursor Down <u><i>n</i></u> Lines         |
| /VT   | Vertical Tab (Up 1 Column)                |

## ZWINTERM Mnemonic Namespace

### **/INIT**(*TermSpace,Attribute*)

Initializes the ZWINTERM Mnemonic Namespace for the current device.

### **/END**

Ends use of the ZWINTERM Mnemonic Namespace.

### **/WOPEN**(*Win,Row,Col,NumRow,NumCol,Flg*)

Creates a new window with identifier *Win* starting at the location specified by the *Row* and *Col* parameters.

### **/WUSE**(*Win*)

Makes the specified window the current window.

### **/WCLOSE**(*Win*)

Closes the window identified by the *Win* parameter.

### **/WSCRON**

Enables automatic display.

### **/WSCROFF**

Disables automatic display.

### **/WDSP**

Displays the contents of the memory buffer on screen.

### **/WUNDSP**

Erases the current window from the screen.

## **/WMOVE(*Row,Col*)**

Moves the location of an existing window.

## **/WT(*Title,Location,Alignment*)**

Specifies a title to appear in the border of the current window.

## **/WTSGR(*Attribute*)**

Sets the attributes for the title that appears in the border of the current window.

## **/WBSGR(*Attribute*)**

Sets or clears attributes associated with the border of the current window.

## **/WDSGR(*Attribute*)**

Sets or clears the default attributes associated with the current window.

## **/WGETCW**

Returns the identifier (i.e., the *Win* value) for the current window.

## **/WREFRESH**

Refreshes the screen.

## **/WCMD(*Type*)**

Erases all or part of the current window.

## **/WCML(*Type*)**

Erases all or part of a line.

## **/WWR(*Mode*)**

Sets the wrap mode.

## **/WSC(*Scroll*)**

Changes the scroll mode.

## ***/WCSGR(Attribute,Location)***

Changes the text attributes.

## ***/CUR(Display)***

Performs various actions on the cursor.

## ***/CUP(Row,Col)***

Positions the cursor to the specified *Row* and *Col* position.

## ***/ED(Type)***

Erases all or part of the current window.

## ***/EL(Type)***

Erases all or part of a line.

## ***/?DR(Char,Repeat)***

Performs line drawing functions.

## ***/SGR(Attribute)***

Sets or clears specified attributes in the current window.

| <b>Erase Attributes</b> |  |
|-------------------------|--|
| 0                       | Erases the window in memory, or the window on screen, or both from and including the current cursor position to the end of the line or the end of the window.  |
| 1                       | Erases the window in memory, or the window on the screen, or both from the beginning of the window or the line to and including the current cursor position.   |
| 2                       | Clears the entire window or line in memory, or the entire window or line on the screen, or both and resets the Blink, Reverse, Bold, and Underline Attributes. |

### Display Attributes

|    |  |
|----|--|
| 0  | Restore Normal Attributes              |
| 1  | Set Bold Attribute for text            |
| 2  | Set Underline Attribute for text       |
| 5  | Set Blink Attribute for text           |
| 7  | Set Reverse Video Attribute for text   |
| 22 | Clear Bold Attribute for text          |
| 24 | Clear Underline Attribute for text     |
| 25 | Clear Blink Attribute for text         |
| 26 | Clear Blink Attribute for text         |
| 27 | Clear Reverse Video Attribute for text |
| 30 | Set foreground color to Black          |
| 31 | Set foreground color to Red            |
| 32 | Set foreground color to Green          |
| 33 | Set foreground color to Yellow         |
| 34 | Set foreground color to Blue           |
| 35 | Set foreground color to Magenta        |
| 36 | Set foreground color to Cyan           |
| 37 | Set foreground color to White          |
| 40 | Set background color to Black          |
| 41 | Set background color to Red            |
| 42 | Set background color to Green          |
| 43 | Set background color to Yellow         |
| 44 | Set background color to Blue           |
| 45 | Set background color to Magenta        |
| 46 | Set background color to Cyan           |
| 47 | Set background color to White          |

### Error Codes

|   |  |
|---|--|
| 0 | Operation completed normally           |
| 1 | Mnemonic Namespace not defined         |
| 2 | Specified Control Mnemonic not defined |
| 3 | Undefined Mnemonic Namespace           |
| 4 | Specified Namespace is not applicable  |
| 5 | Mnemonic Namespace not defined         |
| 6 | No space available to create window    |

## ASCII Character Set

| Decimal | Symbol                  | Pattern |
|---------|-------------------------|---------|
| 0       | NULL                    | C,E     |
| 1       | SOH (Backspace)         | C,E     |
| 2       | STX (Forward Space)     | C,E     |
| 3       | ETX (Write Tape Mark)   | C,E     |
| 4       | EOT (Write Block)       | C,E     |
| 5       | ENQ (Rewind)            | C,E     |
| 6       | ACK (Read Block)        | C,E     |
| 7       | BELL (Read Label)       | C,E     |
| 8       | BS (Write Header Label) | C,E     |
| 9       | HT (Write EOF Label)    | C,E     |
| 10      | LF (Update Tape Status) | C,E     |
| 11      | VT                      | C,E     |
| 12      | FF (Space to Tape Mark) | C,E     |
| 13      | CR (Erase)              | C,E     |
| 14      | SO (Retension)          | C,E     |
| 15      | SI (Space to Data End)  | C,E     |
| 16      | DLE                     | C,E     |
| 17      | DC1                     | C,E     |
| 18      | DC2                     | C,E     |
| 19      | DC3                     | C,E     |
| 20      | DC4                     | C,E     |
| 21      | NAK                     | C,E     |
| 22      | SYN                     | C,E     |
| 23      | ETB                     | C,E     |
| 24      | CAN                     | C,E     |
| 25      | EM                      | C,E     |
| 26      | SUB                     | C,E     |
| 27      | ESC                     | C,E     |
| 28      | FS                      | C,E     |
| 29      | CS                      | C,E     |
| 30      | RS                      | C,E     |
| 31      | US                      | C,E     |

| Decimal | Symbol | Pattern |
|---------|--------|---------|
| 32      | SPACE  | P,E     |
| 33      | !      | P,E     |
| 34      | "      | P,E     |
| 35      | #      | P,E     |
| 36      | \$     | P,E     |
| 37      | %      | P,E     |
| 38      | &      | P,E     |
| 39      | '      | P,E     |
| 40      | (      | P,E     |
| 41      | )      | P,E     |
| 42      | *      | P,E     |
| 43      | +      | P,E     |
| 44      | ,      | P,E     |
| 45      | -      | P,E     |
| 46      | .      | P,E     |
| 47      | /      | P,E     |
| 48      | 0      | N,E     |
| 49      | 1      | N,E     |
| 50      | 2      | N,E     |
| 51      | 3      | N,E     |
| 52      | 4      | N,E     |
| 53      | 5      | N,E     |
| 54      | 6      | N,E     |
| 55      | 7      | N,E     |
| 56      | 8      | N,E     |
| 57      | 9      | N,E     |
| 58      | :      | P,E     |
| 59      | ;      | P,E     |
| 60      | <      | P,E     |
| 61      | =      | P,E     |
| 62      | >      | P,E     |
| 63      | ?      | P,E     |

| Decimal | Symbol | Pattern |
|---------|--------|---------|
| 64      | @      | P,E     |
| 65      | A      | A,U,E   |
| 66      | B      | A,U,E   |
| 67      | C      | A,U,E   |
| 68      | D      | A,U,E   |
| 69      | E      | A,U,E   |
| 70      | F      | A,U,E   |
| 71      | G      | A,U,E   |
| 72      | H      | A,U,E   |
| 73      | I      | A,U,E   |
| 74      | J      | A,U,E   |
| 75      | K      | A,U,E   |
| 76      | L      | A,U,E   |
| 77      | M      | A,U,E   |
| 78      | N      | A,U,E   |
| 79      | O      | A,U,E   |
| 80      | P      | A,U,E   |
| 81      | Q      | A,U,E   |
| 82      | R      | A,U,E   |
| 83      | S      | A,U,E   |
| 84      | T      | A,U,E   |
| 85      | U      | A,U,E   |
| 86      | V      | A,U,E   |
| 87      | W      | A,U,E   |
| 88      | X      | A,U,E   |
| 89      | Y      | A,U,E   |
| 90      | Z      | A,U,E   |
| 91      | [      | P,E     |
| 92      | \      | P,E     |
| 93      | ]      | P,E     |
| 94      | ^      | P,E     |
| 95      | _      | P,E     |

| Decimal | Symbol | Pattern |
|---------|--------|---------|
| 96      | ‘      | P,E     |
| 97      | a      | A,L,E   |
| 98      | b      | A,L,E   |
| 99      | c      | A,L,E   |
| 100     | d      | A,L,E   |
| 101     | e      | A,L,E   |
| 102     | f      | A,L,E   |
| 103     | g      | A,L,E   |
| 104     | h      | A,L,E   |
| 105     | i      | A,L,E   |
| 106     | j      | A,L,E   |
| 107     | k      | A,L,E   |
| 108     | l      | A,L,E   |
| 109     | m      | A,L,E   |
| 110     | n      | A,L,E   |
| 111     | o      | A,L,E   |
| 112     | p      | A,L,E   |
| 113     | q      | A,L,E   |
| 114     | r      | A,L,E   |
| 115     | s      | A,L,E   |
| 116     | t      | A,L,E   |
| 117     | u      | A,L,E   |
| 118     | v      | A,L,E   |
| 119     | w      | A,L,E   |
| 120     | x      | A,L,E   |
| 121     | y      | A,L,E   |
| 122     | z      | A,L,E   |
| 123     | {      | P,E     |
| 124     |        | P,E     |
| 125     | }      | P,E     |
| 126     | ~      | P,E     |
| 127     | DEL    | C,E     |

## Summary of Terminology

### **Actual List (ActList)**

A list of expressions passed as arguments to an Extrinsic function. Arguments are enclosed in parentheses, separated by commas, and appended to the function name.

### **Address**

An integer expression used to specify a particular memory location.

### **ANSI**

The American National Standard Institute.

### **Argument**

An expression which controls the action which is to take place in a function or command.

### **Begin**

Used to specify the starting location within a process. It is used in commands and functions such as FOR, \$EXTRACT, etc.

### **Beginning Line (BLine)**

Used to specify the point where processing will commence. It is utilized in processes such as ZPRINT, ZREMOVE.

### **Bit Parameter (BitParm)**

A parameter used with the Bit functions.

### **Block Number (Block)**

Used to specify the relative block within the database which will be accessed.

## **Command**

The basic unit of work in MUMPS. It is comprised of a command name followed by a space, followed by none, one, or more arguments that the command will act upon.

## **Command Line**

One or more commands grouped together in a single line.

## **Condition**

A truth valued expression.

## **Count**

An integer expression specifying the maximum number of errors that the \$ZVERIFY function will report.

## **Date (Dte)**

A date value in \$HOROLOG format.

## **Decimal**

An integer expression that is the decimal value of the desired ASCII character.

## **Delimiter**

A character recognized by the compiler to separate various pieces of the language so that an unambiguous translation can be made.

## **Destination Variable (DestVar)**

A local or global variable that the MERGE command will insert data into.

## **Device (Dev)**

The designator which specifies where output will be directed.

## **Direction**

An integer value that indicates the order in which a global is to be traversed. A null or 0 value indicates forward and a -1 value indicates reverse.

## **Domain**

An integer expression which specifies the area of memory that is to be accessed.

## **End**

Used to specify the location where the associated process will stop. It is utilized in such processes as a FOR loop, \$EXTRACT.

## **Ending Line (ELine)**

Specifies where processing will terminate. Used in processes such as ZPRINT, ZREMOVE, etc.

## **Entry Reference (EntryRef)**

An argument that specifies a routine or routine line. The EntryRef can be a routine name, the label of a routine line, or the label of a routine line within a specified routine.

## **Expression (Exp)**

A set of variables, constants, and operators which produce a value upon execution. Valid types are number, string or truth-valued expressions.

## **Field**

An integer expression that specifies the size of the string to be returned.

## **First**

An integer expression that specifies the starting occurrence of the substring to retrieve.

## **Format**

A formatting character that indicates the carriage control characters that are to be sent to the device.

## **Global Variable**

A symbolic name assigned to a data value which permanently exists after the terminal session.

## **Increment (Incr)**

Used to specify the value to increase by when executing incremental processes such as a FOR loop.

## **Integer**

Any integer type expression.

## **Last**

An integer expression that specifies the ending occurrence of the substring to retrieve.

## **Length**

An expression that specifies a size.

## **Level**

An integer expression which indicates the level of source code to be retained when doing a ZSAVE.

## **Line Reference (LineRef)**

Used to specify a location within a routine where insertion will take place when using ZINSERT.

## **Local Variable (Local)**

A symbolic name for data which exists during a terminal session or routine execution.

## **Name**

A routine or global variable name.

## **NameSpace (Space)**

A one to fifteen character name of a Mnemonic Name Space.

## **Parameter (Parm)**

One or more expressions, separated by commas or colons and enclosed in parentheses.

## **Partition Size (PartSize)**

The work area assigned to a job. It is used for the local symbol table and the routine being executed.

## **Position**

An integer expression that indicates the relative position within the string.

## **Post Conditional (Post)**

An expression used to conditionally execute a command or an argument of a command.

## **Routine (RoutName)**

A grouping of one or more command lines.

## **Source Variable (SrcVar)**

A local or global variable that the MERGE command will extract data from.

## **String**

Any string type expression.

## **String Literal (String Lit)**

A string of characters enclosed in quotation marks.

## **Substring**

Any string type expression located within a string.

## **Time**

A time value in \$HOROLOG format.

## **Timeout (Time)**

A parameter used to specify a period of time to wait, after which execution should continue.

## **Transaction Parameter (TrnsPrm)**

One or more keyword parameters used to specify information about a transaction.

## **Truth-Valued Expression (TruthExp)**

An expression that evaluates to a Boolean value.

## **Type**

An integer expression indicating the format of the data to be returned.

## **UCI Name (UCIName)**

A 3 character external UCI name.

## **UCI Number (UCINum)**

An internal UCI number.

## **Variable (Var)**

A local or global variable name.

## **Volume Name (Vol)**

A 3 character external Volume Group name.

## **Volume Number (VolNum)**

An internal Volume Group number.





Micronetics Design Corporation  
1375 Piccard Drive  
Suite 300  
Rockville, MD 20850  
U.S.A.  
301-258-2605

Micronetics Europe  
Centro Nord-Sud  
via Campagna  
CH-63934 Bioggio-Lugano  
Switzerland  
41-91-593947

Micronetics Europe  
8 Forest Court  
Oaklands Park  
Wokingham  
Berkshire RG11 2FD  
United Kingdom  
44-734-890500

Micronetics Consulting GmbH  
Hugenottenallee 64  
D-63263 Neu-Isenburg  
Germany  
49-6102-25356